Performance Tools

Tulin Kaman

Department of Applied Mathematics and Statistics

Stony Brook/BNL New York Center for Computational Science

tkaman@ams.sunysb.edu

Aug 23, 2012
Do you have information on exactly where the time is being spent within your applications?
Techniques

How the measurement is obtained?

- **Performance Tool Mechanisms**
  - **Sampling (external, low overhead)**
    - Regularly interrupt the program and record where it is
  - **Instrumentation (internal, high overhead)**
    - Code modification, insert functions into program to record and time events

- The measurements are made
  - **Profiling**: summarizes performance data during execution.
  - **Tracing**: What happens in my code at a given time?
Inclusive and Exclusive Profiles

- Performance with respect to code regions

```
int foo()
{
    int a;
    a = a + 1;
    bar();
    a = a + 1;
    return a;
}
```

Exclusive measurements

Inclusive measurements includes child regions
Performance Steps

1. Assess overall performance
2. Identify functions where most of the time being spent
3. Instrument those functions
4. Measure code performance using hardware counters
5. Identify communication bottlenecks (if Parallel)
How to Detect Performance Problems?

- **Performance:** Count floating-point operation
- **Each architecture has its own theoretical peak performance**

<table>
<thead>
<tr>
<th></th>
<th>theoretical peak performance</th>
<th>Clock speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Blue Gene L</td>
<td>2.8 Gflop/s</td>
<td>700 MHz</td>
</tr>
<tr>
<td>IBM Blue Gene P</td>
<td>3.4 Gflop/s</td>
<td>850 MHz</td>
</tr>
</tbody>
</table>

- **Parallel Performance: Scalability**
  - **Strong Scalability:** Total problem size is fixed while the resources are increased.
  - **Weak Scalability:** Keep the amount of work per core the same. Increase the problem size while increasing the resources.
Performance Tools

- Community Tools:
  - GNU Profiler: tool provided with the GNU compiler
  - Tuning and Analysis Utilities (TAU)
  - PAPI (Performance Application Programming Interface)

- High Performance Computing Toolkit (HPCT) for IBM Blue Gene
  - Message Passing Interface (MPI) Profiler and Tracer tool
  - Xprofiler for CPU profiling
  - Hardware Performance Monitoring (HPM) library
  - Modular I/O (MIO) library
GNU Profiler

- Profiling tool provided with the GNU compiler named GNU profiler (gprof)
- Compile and link with `-g -pg`.
- Enabling profiling is as simple as adding `-pg` to the compile flags
- Run the application
- See files called `gmon.out` created on the working directory
**Flat profile**

- “Flat profile”, which you obtain with gprof command 

  `gprof yourexe gmon.out.0 -p`

### Flat profile:

Each sample counts as 0.01 seconds.

<table>
<thead>
<tr>
<th>% cumulative</th>
<th>self</th>
<th>time</th>
<th>cumulative</th>
<th>self</th>
<th>total</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% time: the percentage of the total running time of the program used by this function.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cumulative seconds: a running sum of the number of seconds accounted for by this function and those listed above it.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>self seconds: the number of seconds accounted for by this function alone.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Call Graph

**gprof yourexe gmon.out.0 -q**

```
<table>
<thead>
<tr>
<th>index</th>
<th>time</th>
<th>self</th>
<th>children</th>
<th>called</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>128.30</td>
<td>1/1</td>
<td></td>
<td>main</td>
<td>[1]</td>
</tr>
<tr>
<td>[3] 63.5</td>
<td>128.30</td>
<td>1/1</td>
<td></td>
<td>dmain</td>
<td>[3]</td>
</tr>
<tr>
<td>0.00</td>
<td>127.47</td>
<td>1/1</td>
<td></td>
<td>main_time_step_loop</td>
<td>[4]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.84</td>
<td>1/1</td>
<td></td>
<td>perform_initialisation</td>
<td>[141]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>end_of_run</td>
<td>[562]</td>
</tr>
<tr>
<td>0.00</td>
<td>127.47</td>
<td>1/1</td>
<td></td>
<td>dmain</td>
<td>[3]</td>
</tr>
<tr>
<td>0.00</td>
<td>127.03</td>
<td>3/3</td>
<td></td>
<td>time_step</td>
<td>[5]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.41</td>
<td>3/3</td>
<td></td>
<td>d_printout</td>
<td>[206]</td>
</tr>
<tr>
<td>0.01</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>Set_up_merge_cell</td>
<td>[557]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.01</td>
<td>9/39</td>
<td></td>
<td>h_scatter_states</td>
<td>[406]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/4</td>
<td></td>
<td>delete_untracked_hyper_surfaces</td>
<td>[661]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>2/4</td>
<td></td>
<td>find_time_step</td>
<td>[802]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>7/243</td>
<td></td>
<td>start_clock</td>
<td>[790]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>6/242</td>
<td></td>
<td>stop_clock</td>
<td>[791]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>5/81895269</td>
<td>2/66</td>
<td>debugging</td>
<td>[160]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>2/66</td>
<td></td>
<td>print_storage</td>
<td>[825]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>2/13069959</td>
<td></td>
<td>debug_print</td>
<td>[267]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/3</td>
<td></td>
<td>g_set_gravity Charts</td>
<td>[1212]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/16</td>
<td></td>
<td>set_iem</td>
<td>[1139]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/10</td>
<td></td>
<td>add_time_clean</td>
<td>[1130]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/28</td>
<td></td>
<td>add_time_start</td>
<td>[1114]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/28</td>
<td></td>
<td>add_time_end</td>
<td>[1113]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/8</td>
<td></td>
<td>d_stop_run</td>
<td>[1164]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/5</td>
<td></td>
<td>add_to_debug</td>
<td>[1188]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/3</td>
<td>time_step</td>
<td>[5]</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>52.10</td>
<td>3/3</td>
<td></td>
<td>parabol_driver</td>
<td>[377]</td>
</tr>
<tr>
<td>0.02</td>
<td>11.97</td>
<td>3/3</td>
<td></td>
<td>Update_merge_cell</td>
<td>[14]</td>
</tr>
<tr>
<td>0.07</td>
<td>0.00</td>
<td>3/3</td>
<td></td>
<td>turbulent_inflow</td>
<td>[552]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/3</td>
<td></td>
<td>average_p_t_r_mach</td>
<td>[864]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>6/66</td>
<td></td>
<td>print_storage</td>
<td>[825]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>9/243</td>
<td></td>
<td>start_clock</td>
<td>[790]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>9/242</td>
<td></td>
<td>stop_clock</td>
<td>[791]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/81895269</td>
<td></td>
<td>debugging</td>
<td>[160]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/4</td>
<td></td>
<td>set_current_chart</td>
<td>[1206]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3/3</td>
<td></td>
<td>no_bc_propagate</td>
<td>[1216]</td>
</tr>
<tr>
<td>0.02</td>
<td>32.07</td>
<td>3/3</td>
<td></td>
<td>time_step</td>
<td>[5]</td>
</tr>
<tr>
<td>[14] 15.9</td>
<td>32.07</td>
<td>3</td>
<td></td>
<td>Update_merge_cell</td>
<td>[14]</td>
</tr>
<tr>
<td>1.50</td>
<td>30.77</td>
<td>3105/3105</td>
<td></td>
<td>oned_turbulence_boundarylayer_solver</td>
<td>[15]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>14175/5897679</td>
<td>260</td>
<td>vel</td>
<td>[280]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>540/119181</td>
<td></td>
<td>g_alloc_state</td>
<td>[600]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>540/1197748</td>
<td></td>
<td>g_set_params</td>
<td>[468]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>648/3101728</td>
<td></td>
<td>g_composition_type</td>
<td>[359]</td>
</tr>
<tr>
<td>1.50</td>
<td>30.77</td>
<td>3105/3105</td>
<td></td>
<td>Update_merge_cell</td>
<td>[14]</td>
</tr>
<tr>
<td>[15] 19.5</td>
<td>27.94</td>
<td>1705725/1705726</td>
<td>IDEAL_dynamic_viscosity_thermalconduct</td>
<td>[16]</td>
<td></td>
</tr>
<tr>
<td>27.94</td>
<td>2.88</td>
<td>1705725/1705726</td>
<td></td>
<td>IDEAL_C_P</td>
<td>[172]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>3105/82700</td>
<td></td>
<td>IDEAL_temperature &lt; cycle 1 &gt;</td>
<td>[28]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>6210/6317</td>
<td></td>
<td>free these</td>
<td>[603]</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>6210/13069959</td>
<td></td>
<td>debug_print</td>
<td>[267]</td>
</tr>
</tbody>
</table>
```
Annotated source listing

- prints out the source code to the application, with notes on how many times each function is called.

```
gprof yourexe gmon.out.0 -A
```
Tuning and Analysis Utilities: TAU

TAU team:
Sameer Shende
Allen D. Malony, Wyatt Spear, Scott Biersdorff, Suzanne Millstein

University of Oregon
http://tau.uoregon.edu
Tuning and Analysis Utilities - TAU

- Performance evaluation tool
- Profiling and tracing toolkit for performance analysis of parallel programs written in C, C++, Fortran, Java and Python
- Support for multiple parallel programming paradigms: MPI, Multi-threading, Hybrid (MPI + Threads)
- Access to hardware counters using PAPI
TAU Configuration

• Each configuration labeled with the options used.
  
  .configure -mpi -arch=bgl -pdt=<pdt-dir> -pdt=xlC 
  -PROFILE(default) / -PROFILECALLPATH/ -MPITRACE/ …

• Each configuration creates a unique Makefile.
  
  • <tau-dir>/bgl/lib for BG/L platform
  • <tau-dir>/bgp/lib for BG/P platform

• Add the bin directory to your path.
  export PATH=/bgl/apps/TAUL/tau-2.18/bgl/bin:$PATH
  export PATH=/bgl/apps/TAUL/tau-2.18/bgl/bin:$PATH
List of TAU’s Makefile on BG/L

Makefile.tau-bgltimers-multiplecounters-mpi-papi-compensate-pdt
Makefile.tau-bgltimers-multiplecounters-mpi-papi-pdt
Makefile.tau-callpath-mpi-compensate-pdt
Makefile.tau-callpath-mpi-pdt
Makefile.tau-depthlimit-mpi-pdt
Makefile.tau-mpi-compensate-pdt
Makefile.tau-mpi-papi-pdt

Makefile.tau-mpi-pdt

Makefile.tau-mpi-pdt-trace
Makefile.tau-multiplecounters-mpi-papi-pdt
Makefile.tau-multiplecounters-mpi-papi-pdt-trace
Makefile.tau-pdt
Makefile.tau-phase-multiplecounters-mpi-papi-compensate-pdt
Makefile.tau-phase-multiplecounters-mpi-papi-pdt

Program Database Toolkit (PDT) provides access to the high-level interface of source code for analysis tools and applications.
TAU Instrumentations

Three methods to track the performance of your application

1. Dynamic instrumentation
2. Compiler based instrumentation
3. Source instrumentation
Dynamic instrumentation through library preloading

- Options: tracking MPI, io, memory, cuda, opencl library calls.
- Default: MPI instrumentation
- Others are enabled by command-line options to `tau_exec`

Example: IO instrumentation is requested.

```bash
$ tau_exec -io ./a.out
$ mpirun -np 4 tau_exec -io ./a.out
```
Compiler Based Instrumentation

- Set environment variables
- Use TAU_MAKEFILE
- Use TAU compiler scripts: `tau_cxx.sh`, `tau_cc.sh`, `tau_f90.sh`
- Set TAU options available to TAU compiler scripts

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-optVerbose</code></td>
<td>Enable verbose output (default: on)</td>
</tr>
<tr>
<td><code>-optKeepFiles</code></td>
<td>Do not remove intermediate files</td>
</tr>
<tr>
<td><code>-optShared</code></td>
<td>Use shared library of TAU (consider when using <code>tau_exec</code>)</td>
</tr>
</tbody>
</table>

Example:

```
$ export PATH =[/path to tau]/[arch]/bin:$PATH
$ export TAU_MAKEFILE=/path to tau/[arch]/lib/[makefile]
$ tau_cc.sh -o hello hello.c
```
Running the Application

- Run the application to generate the profile data files
- Profile data files are generated in the current directory. (DEFAULT)
- The environment variables:
  - PROFILEDIR to store the files in different directory.
  - TAU_VERBOSE to see the steps the TAU measurement systems takes when your application is running
  - TAU_TRACK_MESSAGE to track MPI message statistics

On Blue Gene: In your batch job script file, set the environment variable

```
# @ arguments = -np 16 -env PROFILEDIR=<profile-dir> -exe ...
```
Reducing Performance Overhead with TAU_THROTTLE

- Default rule TAU uses to determine which functions to throttle:
  TAU_THROTTLE_NUMCALLS 100000 (DEFAULT)
  TAU_THROTTLE_PERCALL 10 (DEFAULT)

  Profiling of the function is disabled if the number of function call is more than 100000 times and has an inclusive time per call of less than 10 microseconds.

export TAU_THROTTLE_NUMCALLS=2000000
export TAU_THROTTLE_PERCALL=5
Profiling each event callpath

- Make sure you set the TAU_MAKEFILE
  [path to tau]/[arch]/lib/ Makefile.tau-callpath-mpi-pdt

- Set the environment variable TAU_CALLPATH
- Each event callpath to the depth set by the environment variable TAU_CALLPATH_DEPTH environment variable (default is two)
  - Higher depth introduces more overhead

export TAU_CALLPATH=1 (enables callpath)
export TAU_CALLPATH_DEPTH=100 (defines depth)
Performance Counters

- Performance counters can count hardware performance events such as cache misses, floating point operations.
- PAPI: Performance Data Standard and API package provides a uniform interface to access these performance counters.
- TAU uses PAPI
- Find out which PAPI events are supported in your system.
- Run `papi_avail`
Performance Counters on BG/L microprocessor (PowerPC440)

Vendor string and code : (1312)
Model string and code : PVR-0x5202:0x1891 Serial-R00-M0-N1-C:J16-L01 (1375069073)
CPU Revision : 20034.062500
CPU Megahertz : 700.000000
CPU's in this Node : 1
Nodes in this System : 16
Total CPU's : 16
Number Hardware Counters : 52
Max Multiplexers : 32

<table>
<thead>
<tr>
<th>Name</th>
<th>Derived</th>
<th>Description (Mgn. Note)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_L3_TCM</td>
<td>No</td>
<td>Level 3 cache misses (BGL_UPC_L3_CACHE_MISS_DATA_WILL_BE_REQED_DDR)</td>
</tr>
<tr>
<td>PAPI_L3_LDM</td>
<td>Yes</td>
<td>Level 3 load misses (BGL_UPC_L3_MSHNDLR_TOOK_REQ_RDQ, BGL_UPC_L3_MSHNDLR_TOOK_REQ_RDQ)</td>
</tr>
<tr>
<td>PAPI_L3_STM</td>
<td>No</td>
<td>Level 3 store misses (BGL_UPC_L3_MSHNDLR_TOOK_REQ_WRBUF)</td>
</tr>
<tr>
<td>PAPI_FMA_INS</td>
<td>No</td>
<td>FMA instructions completed (BGL_FPU_ARITH_TRINARY_OP)</td>
</tr>
<tr>
<td>PAPI_TOT_CYC</td>
<td>No</td>
<td>Total cycles (Timebase register (null))</td>
</tr>
<tr>
<td>PAPI_L2_DCH</td>
<td>Yes</td>
<td>Level 2 data cache hits (BGL_UPC_FU_PREF_STREAM_HIT, BGL_UPC_FU_PREF_STREAM_HIT)</td>
</tr>
<tr>
<td>PAPI_L2_DCA</td>
<td>Yes</td>
<td>Level 2 data cache accesses (BGL_UPC_FU_PREF_REQ_VALID, BGL_UPC_FU_PREF_REQ_VALID)</td>
</tr>
<tr>
<td>PAPI_L3_TCH</td>
<td>No</td>
<td>Level 3 total cache hits (BGL_UPC_L3_CACHE_HIT)</td>
</tr>
<tr>
<td>PAPI_FHL_INS</td>
<td>No</td>
<td>Floating point multiply instructions (BGL_FPU_ARITH_MULT_DIV)</td>
</tr>
<tr>
<td>PAPI_FAD_INS</td>
<td>No</td>
<td>Floating point add instructions (BGL_FPU_ARITH_ADD_SUBTRACT)</td>
</tr>
<tr>
<td>PAPI_BGL_OED</td>
<td>No</td>
<td>Floating point Oedipus operations (BGL_FPU_ARITH_OEDIPUS)</td>
</tr>
<tr>
<td>PAPI_BGL_TS_32B</td>
<td>Yes</td>
<td>32B chunks sent in any torus link (BGL_UPC_TS_XM_32B_CHUNKS, BGL_UPC_TS_XP_32B_CHUNKS,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BGL_UPC_TS_YM_32B_CHUNKS, BGL_UPC_TS_YP_32B_CHUNKS, B...)</td>
</tr>
<tr>
<td>PAPI_BGL_TS_FULL</td>
<td>Yes</td>
<td>CLOCKx2 cycles with no token (accum) (BGL_UPC_TS_XM_LINK_AVAIL_NO_VCD0_VCD1_VCD2_VCD3_TOKENS, BGL_U...)</td>
</tr>
<tr>
<td>PAPI_BGL_TR_DPKT</td>
<td>Yes</td>
<td>Data packets sent on any torus channel (BGL_UPC_TR_SNRD_2_VCD0_VCD1_VCD2_VCD3_TOKENS, BGL_U...)</td>
</tr>
<tr>
<td>PAPI_BGL_TR_SEND_2_VCD0_VCD1_VCD2_VCD3_TOKENS</td>
<td>Yes</td>
<td>Data packets sent on any torus channel (BGL_UPC_TR_SNRD_2_VCD0_VCD1_VCD2_VCD3_TOKENS, BGL_U...)</td>
</tr>
<tr>
<td>PAPI_BGL_TR_RECV_0_VCD0_FULL</td>
<td>Yes</td>
<td>CLOCKx2 cycles with tree receiver full (accum) (BGL_UPC_TR_RCV_0_VCD0_FULL, BGL_UPC_TR_RCV_0_VCD1_FULL, BGL_UPC_TR_RCV_1_VCD0_FULL, BGL_UPC_TR_RCV_1_VCD1_FULL)</td>
</tr>
</tbody>
</table>
Performance Counters on BG/P microprocessor (PowerPC450)

PAPI_L1_DCM_idx = 0, /*Level 1 data cache misses */
PAPI_L1_ICM_idx, /*Level 1 instruction cache misses */
PAPI_L2_DCM_idx, /*Level 2 data cache misses */
PAPI_L2_ICM_idx, /*Level 2 instruction cache misses */
PAPI_L3_DCM_idx, /*Level 3 data cache misses */
PAPI_L3_ICM_idx, /*Level 3 instruction cache misses */
PAPI_L1_TCM_idx, /*Level 1 total cache misses */
PAPI_L2_TCM_idx, /*Level 2 total cache misses */
PAPI_L3_TCM_idx, /*Level 3 total cache misses */
PAPI_CA_SNPI_idx, /*Snoops */
PAPI_CA_SHR_idx, /*Request for shared cache line (SMP) */
PAPI_CA_CLN_idx, /*Request for clean cache line (SMP) */
PAPI_CA_INV_idx, /*Request for cache line Invalidation (SMP) */
PAPI_CA_ITV_idx, /*Request for cache line Intervention (SMP) */
PAPI_L3_LDM_idx, /*Level 3 load misses */
PAPI_L3_STM_idx, /*Level 3 store misses */
/* 0x10 */
PAPI_BRU_IDL_idx, /*Cycles branch units are idle */
PAPI_FXU_IDL_idx, /*Cycles integer units are idle */
PAPI_FPU_IDL_idx, /*Cycles floating point units are idle */
PAPI_LSU_IDL_idx, /*Cycles load/store units are idle */
PAPI_TLB_DM_idx, /*Data translation lookaside buffer misses */
PAPI_TLB_IM_idx, /*Instr translation lookaside buffer misses */
PAPI_TLB_TL_idx, /*Total translation lookaside buffer misses */
PAPI_L1_LDM_idx, /*Level 1 load misses */
PAPI_L1_STM_idx, /*Level 1 store misses */
PAPI_L2_LDM_idx, /*Level 2 load misses */
PAPI_L2_STM_idx, /*Level 2 store misses */
PAPI_BTAC_M_idx, /*BTAC miss */
PAPI_PRF_DM_idx, /*Prefetch data instruction caused a miss */
PAPI_L3_DCH_idx, /*Level 3 Data Cache Hit */
PAPI_TLB_SD_idx, /*Translation lookaside buffer shootdowns (SMP) */
PAPI_CSR_FAL_idx, /*Failed store conditional instructions */
To Generate Hardware Counter Profile

- Make sure you set the TAU_MAKEFILE for hardware counter profiling.
  TAU_MAKEFILE=[path to tau]/[arch]/lib/Makefile.tau-multiplecounters-mpi-papi-pdt

- Set the COUNTERx environment variables to specify the type of counter to profile in your job script file
  # @ arguments = -np 1 -env PROFILEDIR=<profile-dir>
  -env "COUNTER1=GET_TIME_OF_DAY COUNTER2= PAPI_L1_DCM \ COUNTER3=PAPI_L1_ICM COUNTER4=PAPI_L1_TCM" -exe ...

- Following subdirectories will be created
  <profile-dir>/MULTI__GET_TIME_OF_DAY
  <profile-dir>/MULTI__PAPI_L1_DCM
  <profile-dir>/MULTI__PAPI_L1_ICM
  <profile-dir>/MULTI__PAPI_L1_TCM
Fast Blue Gene Timers

- Blue Gene systems have a special clock cycle counter that can be used for low overhead timings

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-BGLTIMERS</td>
<td>Use fast low-overhead timers on IBM BG/L</td>
</tr>
<tr>
<td>-BGPTIMERS</td>
<td>Use fast low-overhead timers on IBM BG/P</td>
</tr>
<tr>
<td>-LINUXTIMERS</td>
<td>Use low overhead TSC Counter for wallclock time.</td>
</tr>
<tr>
<td>-CPUTIME</td>
<td>Use usertime+system time instead of wallclock time.</td>
</tr>
<tr>
<td>-PAPIWALLCLOCK</td>
<td>Use PAPI to access wallclock time.</td>
</tr>
</tbody>
</table>
Analyzing Parallel Application

- **pprof (for text based display)**
  - sorts and displays profile data generated by TAU.
  - Execute pprof in the directory where profile files are located.

- **paraprof (for GUI display)**
  - TAU has Java based performance data viewer.
  - Requires Java 1.4 or above, add it to your path.
  - Pack all the profile files into one file. Easy to copy one file to local computer.

  $ paraprof --pack filename.ppk

  - To launch the GUI

  $ paraprof filename.ppk
pprof (Text based display)
Paraprof (for GUI Display)

execute paraprof from the command line where the profiles files are
Function Data and Comparison Windows

File Options Windows Help

Name: MPI_Barrier()
Metric Name: Time
Value: Exclusive
Units: seconds

267.583, 259.453, 251.611, 237.819, 194.684, 161.221, 159.25, 147.137, 73.398, 65.151

mean node 7 node 4 node 2 node 5 node 6 node 3 node 0 node 1 std. dev.

ParaProf: Comparison Window

File Options Windows Help

Metric: Time
Value: Exclusive
Units: seconds

267.583 (181.86%) 147.137

MPI_Barrier()
Performance Counters
Custom Profiling

Selective Instrumentation File

- Specify a list of routines to exclude or include (case sensitive)

BEGIN_EXCLUDE_LIST
F1()
F2()
END_EXCLUDE_LIST

BEGIN_INCLUDE_LIST
int main(int, char **)
F2()
END_INCLUDE_LIST

- Optionally specify a list of files to exclude or include

BEGIN_FILE_EXCLUDE_LIST
f1.c
f2.c
END_FILE_EXCLUDE_LIST

BEGIN_FILE_INCLUDE_LIST
main.c
f2.c
END_FILE_INCLUDE_LIST

What loops account for the most time? How much?

- Generating a loop level profile

```
export TAU_MAKEFILE=$TAULIBDIR/Makefile.tau-mpi-pdt

export TAU_OPTIONS='--optTauSelectFile=select.tau --optVerbose'
```

```
$ cat select.tau

BEGIN_INSTRUMENT_SECTION
loops routine="#"
END_INSTRUMENT_SECTION
```
Question 1: What routines account for the most time?

- Create a Flat Profile

```bash
$ export PATH=/bgl/apps/TAUL/tau-2.18/bgl/bin:$PATH

$ export TAU_MAKEFILE= /bgl/apps/TAUL/tau-2.18/bgl/lib/Makefile.tau-mpi-pdt

$ make CC=tau_cc.sh CXX=tau_cxx.sh F90='tau_f90.sh -qfixed'
```

(Or edit Makefile and change F90=tau_f90.sh)

- In your job script file,

```
# @ arguments = -np 16 -env PROFILEDIR=<profile-dir> -exe …
```

```bash
$ llsubmit tau_app.run
$ cd <profile-dir>
$ paraprof --pack tau_app.ppk
$ paraprof tau_app.ppk
```
Question 2: Who calls MPI_Barrier() function?

- Generate call path profiles

```bash
$ export PATH=/bgl/apps/TAUL/tau-2.18/bgl/bin:$PATH

$ export TAU_MAKEFILE=/bgl/apps/TAUL/tau-2.18/bgl/lib/Makefile.tau-callpath-mpi-pdt

$ make CC=tau_cc.sh CXX=tau_cxx.sh F90='tau_f90.sh -qfixed'
(Or edit Makefile and change F90=tau_f90.sh)

- In your job script file,

```bash
# @ arguments = -np 16 -env PROFILEDIR=<profile-dir> -exe ...

$ export TAU_CALLPATH = 1
$ export TAU_CALLPATH_DEPTH = 100
$ llsubmit tau_app.run
$ cd <profile-dir>
$ paraprof --pack tau_app.ppk
$ paraprof tau_app.ppk
(Windows → Thread → Call Path Relations → Call Graph)
### Answer2: paraprof → Windows → Threads → Call Path Relations

<table>
<thead>
<tr>
<th>Metric Name: Time</th>
<th>Sorted By: Exclusive</th>
<th>Units: seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0E-5</td>
<td>2.0E-5</td>
<td>6/2890</td>
</tr>
<tr>
<td>0.001</td>
<td>0.001</td>
<td>1/2890</td>
</tr>
<tr>
<td>9.5E-6</td>
<td>9.5E-6</td>
<td>5/2890</td>
</tr>
<tr>
<td>0.006</td>
<td>0.006</td>
<td>2890</td>
</tr>
</tbody>
</table>

```c
bool surface_redistribute(Front*, bool*) C {{fredist3d.c} {1358,8}-{1527,1}}
int append_buffer_surface(SURFACE*, SURFACE*, RECT_GRID*, int, int, P_LINK*,
void init_states(INIT_DATA*, INIT_PHYSICS*, CHART*, INPUT_SOLN**, RESTART_DATA,
void add_time_start(int) C {{times.c} {264,8}-{267,1}}

void SG33d(double, Front*, Wave*) C {{gvisc.c} {3060,1}-{4975,1}}
bool f_intfc_communication3d2(Front*) C {{fscat3d2.c} {87,8}-{344,1}}
bool f_intfc_communication3d1(Front*) C {{fscat3d1.c} {439,1}-{732,1}}
bool create_directory(const char*, bool) C {{output.c} {927,8}-{1008,1}}
void communicate_default_comp(Front*) C {{fscat3d1.c} {4710,8}-{4803,1}}
bool h_scatter_states(Wave*, Front*, int*, int) C {{hscatter.c} {139,9}-{175,1}}
void pp_gsync(void) C {{psub.c} {1122,8}-{1148,1}}
void pp_okay_to_proceed(const char*, const char*) C {{psub.c} {148,1}-{158,1}}
MPI_Barrier()
bool debugging(const char*) C {{debug.c} {470,8}-{504,1}}
void debug_print(const char*, const char*, ...) C {{debug.c} {413,8}-{448,1}}
```
τau_cc.sh / τau_cxx.sh / τau_f90.sh

**Compiler wrappers** (PDT instrumentation)

**TAU_MAKEFILE**
- Set instrumentation definition file

**TAU_OPTIONS**
- Set instrumentation options
  - dynamic phase name='name' file='filename' line=start_line_# to line=end_line_#
    - Specify dynamic Phase
  - loops file='filename' routine='routine name'
    - Instrument outer loops
  - memory file='filename' routine='routine name'
    - Track memory
  - io file='filename' routine='routine name'
    - Track IO

**TAU_PROFILE / TAU_TRACE**
- Enable profiling and/or tracing

**PROFILEDIR / TRACEDIR**
- Set profile(trace output directory

**TAU_CALLPATH=1 / TAU_CALLPATH_DEPTH**
- Enable Callpath profiling, set callpath depth

**TAU_THROTTLE=1 / TAU_THROTTLE_NUMCalls / TAU_THROTTLE_PERCALL**
- Enable event throttling, set number of call, percall (us) threshold

**TAU_METRICS**
- List of PAPI metrics to profile
Applying Performance Tools to FRONTIER

- mature, production-quality multiphysics simulation package.
- supports a range of physics, including compressible and incompressible flow, MHD, turbulence models, fluid-structure interactions, phase transitions, and crystal growth.
- DoE Innovative and Novel Computational Impact on Theory and Experiment INCITE, PI: James Glimm
  - 2011 Uncertainty Quantification for Turbulent Mixing
    ANL IBM BG/P 10M core hours
  - 2012 Stochastic (w*) Convergence for Turbulent Combustion
    ANL IBM BG/P 35M core hours

62% efficiency on 163,840 cores
Argonne National Laboratory ALCF Blue Gene P system
We performed mesh refinement on grids of sizes of 24, 192, 1536 million cells, and run them using 1,024, 8,192, and 65,536 cores, respectively, so that the amount of computation for the volume remains constant per core.

The total problem size is fixed while the resources are increased. Scaling starts at 8 racks, which is the smallest configuration with sufficient memory.
Tutorial video and presentations are
http://www.stonybrook.edu/sbccs/tutorials.shtml

Tuning and Analysis Utilities: TAU
http://www.cs.uoregon.edu/Research/tau/home.php